设计实验室 11　　6.01 –Fall 2011

走廊中的机器人

目标： 设计实验 11 为估算机器人在从不确定位置开始沿着走廊移动时的位置奠定了基础。您将：· 探索观测模型和转移模型 · 模拟机器人在走廊中移动的贝叶斯状态估计 · 为构建机器人位置定位的现实世界系统做准备

资源： 本实验应**独立完成**。请运行 athrun 6.01 getFiles。相关文件（位于 ~/Desktop/6.01/designLab11 中）为：· designLab11Work.py：用于模拟彩色走廊状态估计的代码

1 引言

当我们拥有一个具有内部状态的系统，而无法直接观察到这些*状态时,我们就可以考虑状态*估计的问题，即试图基于与系统状态相关的观测来了解系统的内部隐藏状态。这类系统的例子包括：

· 一台复印机，其隐藏状态是其内部机械的状况，我们能采取的行动是复印，而观察到的结果是复印件的质量。

· 一个在走廊中移动的机器人，其中隐藏状态是机器人的位置，我们能采取的行动是向东和向西移动，而观察到的结果是颜色（这些颜色可能并不总是能准确反映墙壁的真实潜在颜色）。

· 一个玩电子游戏的人，其中隐藏状态是此人试图消灭的怪物，不存在明确的行动，而观察到的则是此人正在进行的动作。

状态估计是这样一个过程：接收我们给系统的一系列输入（有时我们称其为动作）以及我们对系统的观测，并计算出系统隐藏状态的概率分布。在接下来的几个实验中，我们将使用基本的**估计状态**来构建一个系统，该系统基于有噪声的声纳和里程计读数来估计机器人的位姿。

要进行概率状态估计，我们需要一个具有三个组成部分的模型：

· 初始分布：关于系统状态的概率分布，告诉我们任何状态为初始状态的概率。

· 一个观测模型：一个条件概率分布，告诉我们给定状态时看到每种可能观测结果的概率。

· 一个转换模型：一种条件概率分布，它告诉我们给定在时间 t 的状态以及在时间 t 的行动，在时间 t + 1 处于每种状态的概率

Some of the software and design labs contain the command athrun 6.01 getFiles. Please disregard this instruction; the same files are available on the 6.01 OCW Scholar site as a .zip file, labeled Code for [Design or Software Lab number].

我们将先对观察和转换模型建立一些熟悉度，然后在一个简单的模拟世界中培养您对这些想法应用的直觉，接着我们将更详细地对真实机器人进行建模，为第 13 设计实验课做准备。

2 走廊世界

我们将对一个机器人在由固定数量的彩色房间组成的一维走廊中上下移动的抽象模拟进行研究。

机器人一开始知道有多少个房间，但不知道自己在哪个房间。

- 机器人知道它在每个房间会做出何种观测结果（颜色）的概率分布。
- 机器人可以尝试在走廊的每个方向移动；它并不总是能完全可靠地移动。如果它靠在走廊的左端并试图向左移动，它就会停留在最左边的房间；同样，如果它在最右边的房间并试图向右移动也是如此。

机器人的目标是确定它所在的房间。我们将使用**状态估计**来解决这个问题。

步骤 1. 使用 -n 启动空闲模式，加载文件 designLab11Work.py 并运行它。（我们不会使用 Soar。这是一个独立的软件。）现在，输入

```
p = makePerfect()
p.run(10)
```

在这一点上，p 是一个小应用程序的实例，它既能模拟机器人在世界中移动，又能显示估计的信念状态。

每个正方形对应于世界中的一个房间。每个房间都有一个“真实”颜色，它决定了机器人在该房间内所做的观察结果的分布。房间的真实颜色显示在每个正方形的外沿。

机器人的*信念状态*是关于它所处房间的概率分布。在窗口中，当前的信念状态以每个模块内方格的内正方形的颜色显示出来，红色值越亮越接近零，蓝色值越亮越接近一。黑色被分配给与均匀分布相关的概率（在本例中为 0.2）。分配给每个房间的概率也会在每个方格中打印出来。

事实上，p 是一个随机状态机的组合，该***随机状态机***模拟机器人 - 世界系统的行为，以及一个*状态机*，该**状态**机基于机器人的动作和观测进行状态**估计**，以在每个时间步长计算新的信念状态。

这种复杂机器的工作方式如下：

- 当 p 初始化时，在每次调用 p.run 的开头：
  1. 状态估计器*将信念状态*初始化为起始信念状态，在这种情况下，这是房间的均匀分布。
  2. 模拟器从*起始*分布中随机为机器人选择一个*初始起始位置*。请注意，*状态估计器并不知道这个真实位置;*它只是用于模拟内部。它也不会以任何方式在窗口中或打印输出中显示。
- 在 p 的每一步上：

1. 模拟器从与机器人当前实际所在的房间相关的观测分布中*生成*一个观测值。这个观测值是一个颜色名称，比如“白色”或“绿色”。
2. 模拟器*会提示用户输入*机器人应采取的*动作*。该动作必须是一个介于-4到4（包括-4和4）之间的整数。
3. 模拟的*机器人移动*的房间数量取决于指定的动作；但如果机器人的运动模型有噪声（我们将在后面详细讨论这意味着什么），那么它不一定能移动指定的确切房间数量。此外，它也不会越过走廊的两端。动作为0时，机器人会尝试停留在当前位置。
4. 状态估计器根据其旧有的信念状态和*观测*结果对其信念状态进行观测*更新*。这种更新取决于观测模型，该模型指定了每种状态的观测结果的概率分布。这个信念状态被打印出来。
5. 状态估计器根据由观测*更新*和指定动作得出的信念状态，对其信念状态进行*转换*更新。该更新取决于转换模型，该模型指定了给定前一状态和动作的下一状态的概率分布。
6. *这些*正方形*被重新绘制*，其颜色和数字反映了新信念状态中的概率。

· 如果您将“quit”作为操作输入，整个机器就会终止。除非您使用一个更大的数字参数调用“run”，否则机器将在 10 步后停止。

如果您想再次运行该机器，最好创建一个新的实例；如果您在旧实例上再次调用“运行”，也是可以的，但在进行一次更新之前，初始的信念显示将不正确。

自我检查 1. 在完美的模拟器中移动机器人。确保您理解代表信念状态的颜色的含义，并且 Python 控制台中打印出的数字是有意义的。如果您有任何疑问，请随时向工作人员询问以获取澄清。

步骤 2. 您刚刚创建的世界具有完美的运动和感知能力。您可以通过以下方式创建一个具有有噪声的运动和感知能力的世界并运行它：

```
n = makeNoisy()
n.run(20)
```

自我检查 2. 在有噪音的模拟器中移动机器人。一定要弄明白颜色意味着什么，并且对可能发生的情况有一个基本的了解。如果有疑问，可以随时向工作人员询问。

2.1 观察模型

阅读材料的第 7.6 节可能有助于理解接下来的章节。

观察模型是一个条件概率分布，指定了机器人处于哪个房间时看到的的颜色：$P(O_t = o_t|S_t = s_t)$。在我们的例子中，o_t 的范围是

```
('black', 'white', 'red', 'green', 'blue', 'purple', 'orange', 'darkGreen',
'gold', 'chocolate', 'PapayaWhip', 'MidnightBlue', 'HotPink', 'chartreuse')
```

并且 s_t 涵盖了机器人可能所在的的所有可能的房间。

如果存在m种可能的观测结果和n种可能的位置，那么通常需要m·n个数字来指定观测模型。在这个问题中，我们假设了一个使模型更加紧凑的假设：机器人的观测结果只取决于它所在的房间的*实际颜色*。也就是说，所有实际为白色的房间在可能的观测结果上的分布相同，所有实际为绿色的房间在可能的观测结果上的分布相同（这通常与实际为白色的房间的观测分布不同）。基于这个假设，我们只需要指定 P（观测*颜色* | *实际颜色*),*然后只要我们知道房间的实际颜色,就*可以找到在任何房间中观察到每种颜色的概率。

$$P(O_t = observedColor \mid S_t = s_t) = P(O_t = observedColor \mid ActualColor = actualColor(s_t))$$

$$P(O_t = observedColor \mid S_t = s_t) = P(O_t = observedColor \mid ActualColor = actualColor(s_t))$$

条件概率分布

$P(O_t =$ *观测到的颜色* | *实际颜色*

它指定了在给定机器人房间的真实颜色的情况下观察到的颜色分布，被称为*观察噪声*分布。我们可以在 Python 中指定一个观察噪声分布，作为一个以实际颜色作为输入并返回观察颜色分布的过程。这里是一个非常简单的例子，它总是观察到真实的颜色。

```
def perfectObsNoiseModel(actualColor):
    return dist.DDist({actualColor: 1.0})
```

现在，给定一个观测噪声分布 obsNoise，例如 perfectObsNoiseModel，我们可以构建整个观测模型（给定机器人位置的观测颜色上的条件概率分布），如下所示:

```
def makeObservationModel(hallwayColors, obsNoise):
    return lambda loc: obsNoise(hallwayColors[loc])
```

在这里，hallwayColors 是一个列表，指定了走廊中每个位置的真实颜色，loc 是一个表示机器人位置的整数，obsNoise 是给定实际颜色的情况下观测到的颜色的有条件分布。该过程返回一个有条件的概率分布，这是一个以位置作为输入并返回观测颜色分布的过程。

我们一直使用的示例世界是通过以下方式指定的

```
standardHallway = ['white', 'white', 'green', 'white', 'white']
```

鉴于这些程序，我们可以用以下方式指定完美观测的观测模型:

```
perfectObsModel = makeObservationModel(standardHallway, perfectObsNoiseModel)
```

第三步

Wk.11.1.1 Do this tutor problem on defining observation models.

Check Yourself 3. Just to be sure you understand the observation models, consider a world with two rooms: room R_0 is actually green and room R_1 is actually white.

- With a perfect sensor, what is the probability distribution over observations for each room?

- With `whiteEqGreenObsDist`, what is the probability distribution over observations for each room?

- With `whiteVsGreenObsDist`, what is the probability distribution over observations for each room?

第四步。现在考虑一个更大的测试世界，称为测试走廊。两端的房间是真正的彩色“巧克力”。其他的房间要么是绿色的，要么是白色的。

尝试使用你的 whiteEqGreenObsDist 和 whiteVsGreenObsDist 函数，看看信念状态会发生什么变化。将导师给出的这些函数的定义粘贴到你的 designLab11Work.py 文件中。使用如下所示的完美运动模型。**确保将类似于 whiteEqGreenObsDist 这样的函数作为下面的第三个参数**。（变量 actions、standardDynamics 和 perfectTransNoiseModel 已经在 designLab11Work.py 中为你定义好了）

```
w = makeSim(testHallway, actions,
            <your observedColor given actualColor model>,
            standardDynamics, perfectTransNoiseModel)
w.run(50)
```

2.2 状态转换模型

状态转移模型是在给定时间 t 的状态和所选动作 a 的情况下，时间 t + 1 的状态的条件概率分布。也就是说，$P(S_{t+1} = s_{t+1}|S_t = s_t, A_t = a_t)$。系统的下一个状态既取决于它之前的位置，也取决于所采取的动作。如果您将此模型写成矩阵形式，它将会非常大：如果 n 是世界的状态数量，m 是动作数量，那么它的大小将是 mn^2。

通常，转移模型可以用更稀疏或更系统的方式描述。在这个特定的世界中，机器人可以尝试向右或向左移动一定数量的房间，或者停留在当前位置。我们将假设机器人试图向给定方向移动时所犯的错误类型不取决于机器人实际所在的位置（除非它位于世界的边缘），并且我们将进一步假设向大多数状态的转移概率为零（例如，机器人没有机会传送到走廊的另一端）。这将使我们能够更紧凑地描述转移模型。

我们将通过首先定义一个*动态*过程来开始定义转换模型，例如下面的标准动态过程。动态过程是一个程序，它返回在走廊中从位置 loc 采取动作 act 所得到的标称新位置，走廊长度为 hallwayLength 个位置。这在其他问题中会很有用。机器人可以采取的可能动作为：-4，...，-1，0，1，...，4。因此，我们可以将机器人的动作加到其当前位置上，以得到其标称新位置，但我们必须确保它不会移动到世界边缘之外，所以我们使用 util.clip 来防止值低于 0 或高于 hallwayLength - 1 。

```
def standardDynamics(loc, act, hallwayLength):
    return util.clip(loc + act, 0, hallwayLength-1)
```

接下来，我们定义一个与位置无关的*噪声模型*。它根据给定动力学下采取行动所产生的标称位置，返回一个关于可能的结果位置的分布。最简单的噪声模型假设转换是完美的，因此结果位置将是标称位置。

```
def perfectTransNoiseModel(nominalLoc, hallwayLength):
    return dist.DDist({nominalLoc : 1.0})
```

最终，我们需要构建一个完整的转换模型，其形式为$Pr(S_{t+1} | S_t, A_t)$。由于它取决于两个变量，我们将使用嵌套过程（表示条件分布）来表示它。因此，我们可以将其视为类似于$Pr(S_{t+1} | S_t | A_t)$，或者，作为一个过程，它接受一个a_t并返回一个过程，该过程接受一个s_t并返回一个关于S_{t+1}的分布。

以下是转换模型的基本形式，它包含两个过程，即动态过程和噪声模型，以及一个表示走廊长度的整数。

```
def makeTransitionModel(dynamics, noiseDist, hallwayLength):
    return lambda act: lambda loc: noiseDist(dynamics(loc, act, hallwayLength),
                                             hallwayLength)
```

在标准动力学下，针对我们的标准走廊构建了一个完美的过渡模型，如下所示：

```
perfectTransModel = makeTransitionModel(standardDynamics, perfectTransNoiseModel, 5)
```

第 5 步

第 11 周 1.2 节 做这个关于定义转换模型的辅导题。

第 6 步。现在考虑一个只有白色房间的测试世界，并且存在嘈杂的转换和观测。我们将用初始信念状态初始化状态估计器，该状态将概率 1 分配给位置 7（当然，将概率 0 分配给所有其他位置）。变量 sterile 指定了一个由 16 个白色房间组成的走廊。您可以使用以下方式创建这个世界和状态估计器：

```
w = makeNoisyKnownInitLoc(7, sterile)
w.run(50)
```

连续多次选择动作 0 进行实验。会发生什么？当你驾驶机器人四处移动时又会发生什么？

核对 1。 白色与绿色观测距离（whiteEqGreenObsDist）和白色与绿色观测差异（whiteVsGreenObsDist）是如何观测的？
模型（来自《自我检查 3》）比较：
· 一个完美的传感器模型
· 一种无论处于哪个房间总是读取“黑色”的传感器
向一名工作人员展示从第 6 步开始的世界中的嘈杂动态。
并解释它为什么会有这样的表现。

3 走廊世界中的状态估计

一定要非常仔细地阅读阅读材料的第 7.5 - 7.7 节。

我们将通过手动做一些状态估计的数值示例来建立状态估计的直觉。这些是我们期望你们能够在测验和考试中能够完成的问题类型。

第 7 步

第 11 周 1.4 在走廊世界中做这个关于状态估计的问题。

第八步

第 11 周 1.5 节 在走廊世界中做这个关于状态估计的问题。

4 准备本地化

在设计实验 13 中，我们将构建一个能让机器人“定位”自身的系统：也就是说，根据世界中的障碍物地图以及做出局部动作的能力来估计其在世界中的位置。

声纳读数。这些问题构建了定位系统的重要概念和组成部分。

第 9 步

第 11 周 1.6 节 理解声纳几何。

第 10 步

第 11 周 1.7 计算机器人某一姿势的理想声纳读数。

6.01SC 《电气工程与计算机科学导论》 2011 年春季